

Packaging for Debian

Cedric Staniewski

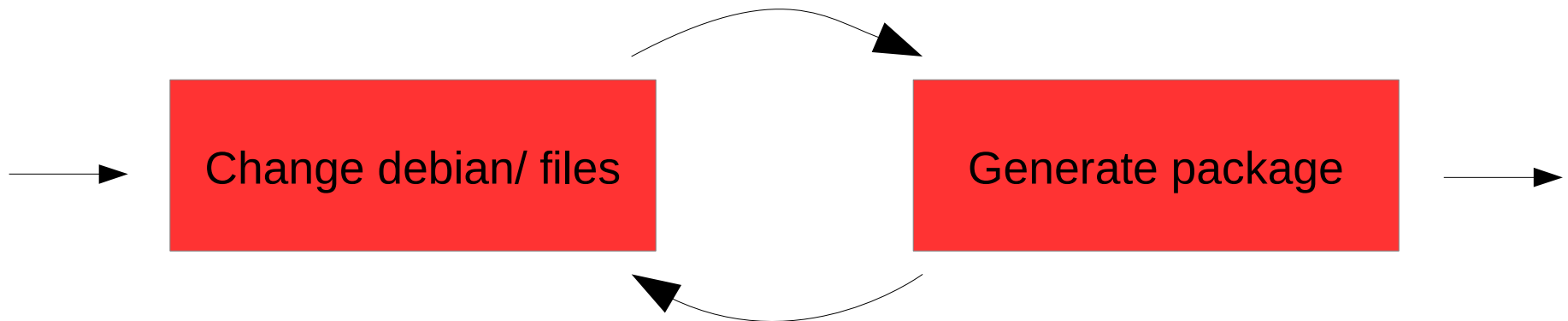
- Motivation
- Creating Debian packages / Useful packaging tools
- PredictProtein

- Software often depend on libraries or external programs
- Two solutions:
 1. Distribute the software in a bundle with its dependencies
 2. Somehow define the dependencies along with the software

Advantages of solution 2

- Libraries may be used by a lot of software, e.g. libpng
 - only a single copy on the harddrive and in the ram
- Updates can be applied globally
 - Huge advantage in the case of security fixes

1. Find a unique package name
2. Manage to compile the source
3. Setting up the debian specific configuration
4. Generate the initial debian package
5. Adjust accordingly to apply to the policies and rebuild
6. Put it into a repository



Make sure the required programs are installed:

```
$ apt-get install build-essential
```

Compile it

```
$ tar xaf grep-2.12.tar.xz
```

```
$ cd grep-2.12
```

```
$ ./configure
```

```
$ make
```

But often, not so straight forward, e.g. missing or old dependencies on the system.

Generate the initial debian files

```
$ tar xaf grep-2.12.tar.xz  
$ cd grep-2.12  
$ dh_make -f ../grep-2.12.tar.xz
```

Afterwards, check and adjust the files under debian/

- control
- rules: check for available debhelper scripts

Eventually: `dpkg-buildpackage`

- Used by 98% of debian packages
- Called in *debian/rules*
- Provides a bunch of scripts for common packaging tasks
 - Icons files / Mime files / Menu files
 - Info / man pages
 - Udev rules
 - ...

Default debian/rules by dh_make

```
#!/usr/bin/make -f
%:
    dh $@
```

- Often, upstream releases contain known bugs or security issues
- The upstream release needs to be adjusted to match packaging policies

→ Patches

- Patch management toolkit
- Used to manage a series of patches (add, edit, remove)
- Patches organized in a stack
- Similar to git, but no history is stored
- Integrated into dpkg since Squeeze

```
$ quilt new <patchname>
```

```
$ quilt add <filenames>
```

```
[make some changes to the added files]
```

```
$ quilt refresh
```


```
$ quilt pop -a
```

- For quality assurance
- Applied on the generated debian package
- Prints out bugs and policy violations in the package, e.g. files installed to /usr/local/

Example output:

```
$ lintian libc5_5.4.38-1.deb
W: libc5: old-fsf-address-in-copyright-file
W: libc5: shlib-without-dependency-information usr/lib/libgnumalloc.so.5.4.38
W: libc5: shlib-without-dependency-information lib/libc.so.5.4.38
W: libc5: shlib-without-dependency-information lib/libm.so.5.0.9
E: libc5: shlib-with-executable-bit lib/libc.so.5.4.38 0755
E: libc5: shlib-with-executable-bit lib/libm.so.5.0.9 0755
E: libc5: shlib-missing-in-control-file libgnumalloc usr/lib/libgnumalloc.so.5.4.38
```

- Packages in the repositories are automatically checked

Last updated:	Tue, 03 Jul 2012 12:48:04 +0000
Archive timestamp:	Tue Jul 3 08:33:20 UTC 2012
Distribution:	sid
Archive area:	main, contrib, non-free
Architecture:	i386
Maintainers:	2187 (-2)
Source packages:	13647 (-1)
Binary packages:	29870 (-12)
µdeb packages:	87 (+0)
 Errors:	2461 (-3)
 Warnings:	82833 (+79)
 Info tags:	102013 (-22)
 Pedantic tags:	49645 (-15)
 Overridden tags:	31239 (+1)
 Experimental tags:	42240 (-3)
Lintian version:	2.5.9

- Administrators of the Debian FTP archive
 - Joerg Jaspert
 - Mark Hymers
 - Torsten Werner
- Their tasks include
 - Keep archive up and running
 - Make sure the files are legal
 - Support the teams working with it, like the release or the security teams
 - Maintain dak (Debian Archive Kit); used to manage the repositories

- Generally, packages should be built in chroots
- Prevents “filesystem corruptions”
- Only specified dependencies are pulled in

PredictProtein

- Input: single FASTA-formatted sequence



Table 1. Methods used by PP

Method	Task	Main author(s)	References
<i>Database</i>			
SWISS-PROT*	Annotated protein sequences	A. Bairoch (SIB) and R. Apweiler (EBI)	(28)
TrEMBL*	Raw protein sequences	R. Apweiler (EBI)	(28)
PDB*	Protein structures	P. Bourne (UCSD)	(29)
BIG	Non-redundant combination of SWISS-PROT, TrEMBL, PDB	D. Przybylski (Columbia)	(7)
<i>Alignment</i>			
MaxHom	Dynamic programming, multiple alignment	R. Schneider (LION) and C. Sander (Sloan Kettering)	(6)
BLASTP*	Pairwise alignment	S. Karlin and S. F. Altschul (NCBI)	(4)
PSI-BLAST*	Profile based alignment	S. F. Altschul (NCBI)	(5)
TOPITS	Prediction-based threading	B. Rost	(11,30,31)
<i>Protein domains and unusual regions</i>			
ProDom*	Structural domain-like regions	F. Corpet, F. Servant, J. Gouzy and D. Kahn (Toulouse)	(32)
SEG*	Low-complexity regions	J. C. Wootton and S. Federhen (NCBI)	(18)
NORSp	Floppy regions	J. Liu and B. Rost	(19,33)
<i>Protein structure</i>			
PHDsec	Secondary structure	B. Rost	(12,34,35)
PHDacc	Solvent accessibility	B. Rost	(12,36)
PHDhtm	Membrane helices	B. Rost	(12,37,38)
PROFsec	Secondary structure	B. Rost	(13)
PROFacc	Solvent accessibility	B. Rost	unpublished
GLOBE	Globularity	B. Rost	unpublished
COILS	Coiled-coiled regions	A. Lupas (Tübingen)	(39)
CYSPRED*	Disulfide-bonds	P. Fariselli and R. Casadio (Bologna)	(15)
ASP	Structural switches	M. Young and S. Highsmith (Sandia)	(17)
<i>Protein function</i>			
PredictNLS	Nuclear localisation signals	R. Nair, M. Cokol and B. Rost (Columbia)	(40,41)
PROSITE*	Functional sequence motifs	K. Hofmann, P. Bucher and A. Bairoch (SIB)	(10)
<i>Tools integrated into PP</i>			
Mview*	HTML alignment viewer	N. Brown	(42)
ESPrpt*	Ready-to-publish output for sequence alignments and secondary structure	P. Gouet and E. Courcelle (IPS Toulouse)	(43)

(2003)

Impressive list of dependencies:

blast2	bioperl	coiledcoils
disulfinder	libnhgri-blastall-perl	libnhgri-blastall-perl
librg-liu-bundle-perl	librg-utils-perl	loctree
lowcompseg	metadisorder	norsnet
profasp	profbval	profchop
profcon	profdisis	profglobe
profisis	profphd	proftmb

Thanks